REMARKS

Receipt of the Office Action of December 23, 2009 is gratefully acknowledged.

Claims 6 - 9 have been reexamined and finally rejected under 35 USC 102(b) as being anticipated by Clift et al (U.S. Patent No 6,633,970).

This rejection is respectfully traversed.

In paragraph 5 of the latest Office Action, the Examiner states that "In response to Applicant's argument, the Examiner submits that Clift discloses 'register, array'. Register and array in Clift is equivalent to 'stack'." As is usual, in the context of the present application, the word of "stack" means a last-in first-out (LIFO) memory, namely that of "stack machine". The above-cited Examiner's submission is incomprehensible.

Clift's system comes under the category of a register machine not a stack machine, because each entry of the RAT primary/shadow array 110/112 corresponds to a predetermined logical register (Clift: Fig. 2, Fig. 4, column 5 lines 11-25 and column 11 lines 16-29). Of course, a register machine can emulate a stack machine. Clift et al., however, do not disclose any stack management system, and cannot, it is respectfully submitted therefore, anticipate claims 6-9.

In paragraph 3 of the Office Action, the Examiner also states that "the recitation 'stack management system' has not been given patentable weight because the recitation occurs in the preamble. A preamble is generally not accorded any patentable weight where it merely recites the purpose of a process or the intended use of a structure, and where the body of the claim does not depend on the preamble for completeness but, instead, the process steps or structural limitations are able to stand alone." The Applicant is convinced that a better understanding of the matter will resolve the above-cited Examiner's argument.

Then, the way the computer system disclosed in the present application deals with stack instructions (instructions for the stack machines) is reviewed below.

The computer system disclosed in the present application can operate in either of the two modes: the stack mode and the register mode. On page 8, line 11 - page 11 line 7 of the specification of the present application, how stack instructions are converted into instructions for the stack mode is illustrated. Specifically, five stack instructions { LOAD <4>; DUP; LOAD <1>; MUL; SWAP } are converted into Instruction_1, and five stack instructions { LOAD <2>; SWAP; DIV; ADD; STORE <5> } are converted into Instruction_2. As shown on page 11 at the top, contents of the Op (operations) and SM (state modification) fields of Instruction_1 and Instruction_2 are as follows:

```
Instruction_1
  Op{ load f1, <4>;  add f2, f1, 0;  load f3, <1>;  mul f4, f2, f3 }
  SM{ +2:  f4, f1 }
Instruction_2
  Op{ load f1, <2>;  div f2, f1, s0;  add f3, s1, f2;  store <5>, f3 }
  SM{ -2: }
```

(f1 - f4 represent addresses of the data-file entries to be allocated to hold result data).

The state-modification field content of Instruction_1, namely SM{ +2: f4, f1}, implies that the stack is to be grown by two elements, which are to correspond to f4 and f1. And, the state-modification field content of Instruction_2, namely SM{ -2: }, implies that the stack is to be shrunk by two elements.

On page 28 line 1 - page 32 line 25 of the specification, an example action of processing two instructions, Instruction_1 and Instruction_2, is described.

In the example action, when the computer system is in such a state as shown in Fig. 8, Instruction_1 is issued and the advanced mapping file (AMF 3a) is so modified that p26 and p51, replacing f1 and f4, are respectively entered into the entries of address 0 and 1, and for the parts below, contents of the AMF entries are shifted down by the amount of stack growth (2 entries). (The contents of the AMF entries of address 0, 1, ... shown in Fig. 8 are moved into the AMF entries of address 2, 3, ..., respectively.) The state of the computer system right after issue of Instruction_1 is shown in Fig. 9. The equivalent action of processing Instruction_1 in a traditional system is illustrated in Application No.10/344,698: page 38 lines 13-23, Fig. 13 and Fig. 14.

In the next cycle, Instruction_2 is issued. As only a negative growth of the stack (-2) is indicated in the SM field, contents of the AMF entries are shifted by this amount. (The contents of the AMF entries of address 2, 3, ... shown in Fig. 9 are moved into the AMF entries of address 0, 1, ..., respectively.) The state of the computer system right after issue of Instruction_2 is shown in Fig. 10. The equivalent action of processing Instruction_2 in a traditional system is illustrated in Application No.10/344,698: page 39 line 15 - page 40 line 1, Fig. 14 and Fig. 15.

As illustrated above, AMF 3a is to be manipulated in a peculiar manner in the stack mode of the system disclosed in the present application (page 23 line 21 - page 24 line 21).

In the stack mode of the system disclosed in the present application, the entry of address 0 of AMF 3a is always to correspond to the top of the stack, and the entry of address n of AMF 3a is to correspond to the (n+1)th element of the stack (page 14 line 23 - page 16 line 13 and Fig. 3). On the other hand, traditional look-ahead stack management systems employ the circular buffer technique, which requires two pointers to indicate the top and the bottom.

In regard to claims 6-9, which are included in the RESPONSE filed on August 11, 2009, consider further the following, which were basically included in the REQUEST FOR RECONSIDERATION WITH AMENDMENT filed on March 6, 2009:

First, it should be noted that the data file (DF 6) and the advanced mapping file (AMF 3a) noted in the specification are respectively referred to as "data storing means" and "look-ahead mapping means" in claims 6-9.

As per claim 6, the phrase "for each entry of said look-ahead mapping means that is to hold an entry address in said data storing means allocated to an operand stack element" in lines 8-10 is inserted in order to exclude entries that is to be below the bottom (shaded area of AMF 3a in Fig. 3), and "the entry" in line 10 refers to "each entry" in line 8.

As per claim 7, the phrase "for each entry of said look-ahead mapping means holding an entry address in said data storing means allocated to an operand stack element" in lines 8-10 is inserted in order to exclude entries that is below the bottom right before the modification. The phrase "if the entry of said look-ahead mapping means is to hold an entry address in said data storing means allocated to an operand stack element" in lines 10-12 is inserted in order to exclude cases where "the entry", which refers to "each entry" in line 8, is to fall below the bottom right after the modification. The phrase "whose value is one of:

held and to be held in the entry of said data storing means indicated by the address held in the entry of said look-ahead mapping means" in lines 13-15 qualifies "the operand stack element" in line 13; and the phrase "indicated by the address held in the entry of said look-ahead mapping means" in lines 14-15 qualifies "the entry of said data storing means" in line 14. And, "the entry" in lines 14-15 also refers to "each entry" in line 8. Hence, "the number of operand stack elements over the operand stack element whose " is to be unchanged. Since "look-ahead mapping means" indicates the look-ahead state of the system, the entry of the data storing means indicated by an address held in the look-ahead mapping means may or may not hold the value at the time of the modification. So, the expression of "one of: held and to be held" is adopted.

Claims 8 and 9 are respectively identical to claims 6 and 7 except that "entry"/"entries" is changed to "register"/"registers", and accordingly, "(entry) address" is changed to "(register) number".

The circuit for making a modification on a look-ahead mapping means can be streamlined by adopting the look-ahead stack management system according to claims 6-9. It is not possible to achieve this result with a traditional stack management system equipped with a circular buffer. And, the look-ahead mapping means can have a non-power-of-two number of entries (registers) for stack management.

It is clear that under 35 USC 102, a single reference must disclose each and every positively recited limitation. Inferences are not permitted. The teaching must be unambiguous. That is lacking here, and consequently6, 35 USC 102 does not apply.

In view of the foregoing, reconsideration is respectfully requested and claims 6 - 9 found allowable. If questions still remain, the examiner is urged to contact the undersigned to arrange for an interview for the purpose of resolving any further questions to thereby advance prosecution of this application.

Respectfully submitted,
BACON & THOMAS, PLLC

Date: April 23, 2010

[illegible]

Felix J. D'Ambrosio
Registration No: 25,721

Customer Number *23364*
BACON & THOMAS
625 Slaters Lane, Fourth Floor
Alexandria, Virginia 22314
Phone: (703) 683-0500
S:\Producer\fjd\CLIENTS\Kyomei Int'l Patent & TM Office\SEKI3006\April 23, 2010 Response.wpd